(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 752 786 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
08.01.1997 Bulletin 1997/02

(51) Int. Cl.[6]: **H04N 7/167**

(21) Application number: **96110363.7**

(22) Date of filing: **27.06.1996**

(84) Designated Contracting States:
DE ES FR GB IT

(30) Priority: **07.07.1995 US 499280**

(71) Applicant: **THOMSON CONSUMER ELECTRONICS, INC.**
Indianapolis, IN 46206 (US)

(72) Inventors:
- **Rohatgi, Pankaj**
  Sunnyvale, CA 94086 (US)
- **Dureau, Vincent**
  Venice, CA 90291 (US)

(74) Representative: **Hartnack, Wolfgang, Dipl.-Ing. et al**
Deutsche Thomson-Brandt GmbH
Licensing & Intellectual Property,
Göttinger Chaussee 76
30453 Hannover (DE)

(54) Apparatus and method for authenticating transmitted applications in an interactive information system

(57) An executable interactive program is combined with audio/video data for transmission. The program is divided (40, 41) into modules, similar to computer files, and a Directory Module is created which links program modules. Security for the executable application is provided (52,53,54) by attaching a signed certificate to respective Directory Modules. The integrity of respective modules is monitored by hashing (46) respective modules and including (50) respective hash values in the Directory Module. A hash value of the Directory Module containing the other hash values is generated (46), encrypted (51) and attached (53,54) to the Directory Module. At a receiver the certificate is decoded and checked for authenticity of provider. If the certificate is authentic the program may be executed but only if receiver generated hash values of respective program modules are identical to corresponding hash values contained in the Directory Module.

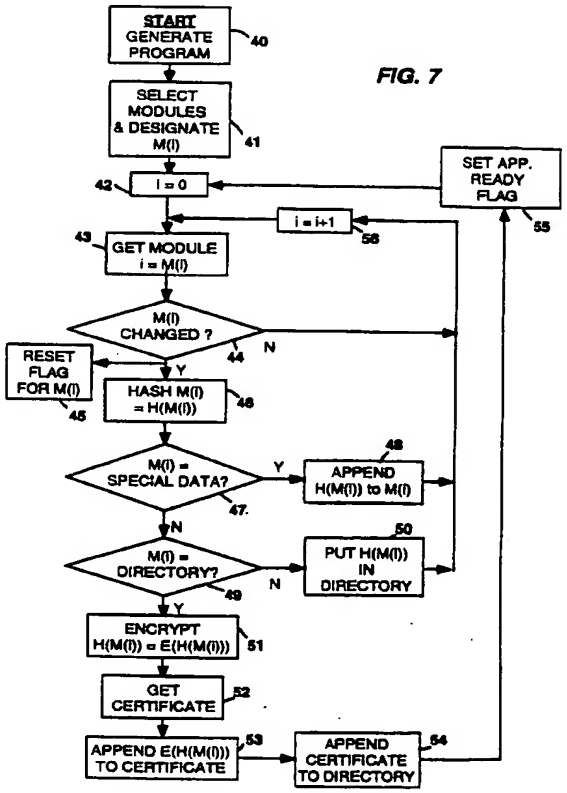

FIG. 7

EP 0 752 786 A1

respective AVI programs are applied in transport packet form to respective input ports of a channel multiplexer 28. The channel multiplexer 28 may be of known construction for equally time division multiplexing respective packet signals into a single signal stream or it may be a statistically controlled multiplexer. The output of the multiplexer 28 is coupled to a forward error coding (FEC) and signal interleaving apparatus 31, which may include Reed-Solomon and Trellis encoders. The output of the FEC 31 is coupled to a modem wherein the multiplexed signal is conditioned for application to, for example a satellite transponder. An exemplary statistically multiplexed packet signal is illustrated in FIGURE 2, and an exemplary format for respective packets is illustrated in FIGURE 3.

AVI formation is controlled by a system program controller 5. Program controller 5 may have a user interface by which particular programs and respective program signal components are selected. The program controller assigns respective SCID's for respective audio, video and interactive components of respective programs. The presumption is made that respective receivers will access a program guide to determine which SCID's associate AVI program components, and then select transport packets from the transmitted signal stream containing the associated SCID's. The audio, video and interactive components are assigned different SCID's so that one or more of the components of one AVI program may conveniently be utilized in the formation of alternate AVI programs. Using differing SCID's also facilitates editing audio from one program with video from another etc.

A given AVI program may include a variety of signal component sources. Shown in FIGURE 1 are an interactive component source 10, a video source 17, and first and second audio sources 20 and 23 (bilingual audio). The controller 5 communicates with respective sources for time management and/or enabling functions. The source of video signal 17 is coupled to a video signal compression apparatus 18, which may compress signal according to the video compression standard promoted by the Moving Pictures Experts Group (MPEG). Similarly the respective audio signals from the sources 20 and 23 are applied to respective compression apparatus 21 and 24. These compression apparatus may compress the respective audio signals according to the audio compression standard promoted by the Moving Pictures Experts Group (MPEG). Associated audio and video signals compressed according to the MPEG protocol are synchronized with the use of presentation time stamps (PTS), which are provided by a timing element 15. For insight into how the audio and video may be temporally related, the reader's attention is directed to INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, ISO/IEC JTC1/SC29/WG11; N0531, CODING OF MOVING PICTURES AND ASSOCIATED AUDIO, MPEG93, SEPTEMBER, 1993.

The compressed audio and video signals are applied to transport packet formers or processors 19, 22 and 25. Audio and video transport packet processors are known and will not be described. Suffice it to say that the packet processors divide the compressed data into payloads of predetermined numbers of bytes and attach identifying headers including SCID's, as indicated in FIGURE 3. For detailed information on a video signal transport packet processor, the reader is directed to United States Patent No. 5,168,356. The packet processors are coupled to the packet multiplexer which time division multiplexes the signal components. The transport packet processor may include a buffer memory for temporarily storing packetized data to accommodate the multiplexer servicing other components. The packet processors include PACKET READY signal lines coupled to the multiplexer to indicate when a packet is available.

Interactive programs are created, via known techniques, by a programmer operating the interactive component source or programming element 10, which may be a computer or personal computer (PC). In forming the application the programming element 10 interfaces with a memory 11 and memory controller 12, and the completed application is stored in the memory 11. After completion, the application may be condensed or translated to some native code to conserve signal bandwidth.

In the formation of the application, portions of the programs are formatted into modules, transmission units and packets as shown in FIGURE 4. The packets are similar in form to the transport packets described above. A transmission unit consists of a plurality of transport packets. Each transmission unit includes a header packet which includes information descriptive of the transmission unit contents, and a plurality of basic packets each of which includes a portion of the codewords of the application. A module is divided into transmission units to facilitate interleaving of information from different modules. Preferably, interleaving of transmission units is permitted, but interleaving of transport packets from different transmission units is not. For a more detailed description of the formation of an application and transmission units etc. the reader is referred to United States Patent No. 5,448,568.

Modules are similar to computer files, and are of different types. A first type of module is a Directory Module which contains information to interrelate respective transmission units and modules as an application. A second module type is a Code Module which comprises executable code necessary to program a computing device at a receiver to perform or execute the application. A third module type is a Data Module. Data Modules include non-executable "data" used in the execution of the application. Data Modules tend to be more dynamic than Code Modules, that is Data Modules may change during a program while Code Modules generally do not. A fourth type of module is designated a Signal Module. This module is a special packet including information to trigger receiver interrupts, which may be utilized for synchronization of e.g., video with application program features, or to suspend operation of an application, or to re-enable operation of a program etc. Synchronization is effected via inclusion of presentation time stamps. Data, Directory, Code and Signal Modules are examples of the PC-data.

The respective modules may be error coded by the programming element 10. For example the entire module may

else if A> B

4. compute C = A- B
5. compare C < N; If C > N quit, signature cannot check,

else if C < N

6. compute E = C times S
7. compute F = Q2 times N
8. Compare E > F, If E < F quit, signature cannot check,

else if E > F

9. compute G = E - F
10. Compare G = D , if not, signature does not check.

Note that all arithmetic operations are simple multiplications or subtractions. The detection of an erroneous signature may occur at steps 3, 5, 8 or 10. If an erroneous signature is detected at steps 3 or 5, very little computational time has been expended.

The AVI receivers are provided with the public keys (and desirably helping quotients Q1 and Q2) of the respective system providers for decrypting signed certificates included with PC-data to determine program authenticity. If program authenticity is not confirmed, the received application is immediately dumped from the receiver. Central to such a security system is the assignment of unique identifiers, ID's, to application providers and the system controller or server. The system controller allocates unique e.g. 32-bit ID's to each trusted AVI application provider and also issues a certificate for the application provider's public key. The certificate is essentially the system controller's digital signature on the application provider's public key and includes fields such as the expiration date of the certificate, the ID of the provider, and a limit on the amount of storage an application with this ID can use in the file system of the receivers. The system controller may employ a plurality of different private-public key pairs, and the certificate may include flags to designate which of the plurality of public keys the receiver should use to decrypt respective certificates.

A certificate for application providers will nominally include:
CERTIFICATE_FLAGS (which identify the certificate type and may include the system controller's public key flag);
PROVIDER_ID, (which indicate length of provider ID);
PROVIDER_EXPIRE, (which indicates application lifetime);
PROVIDER_AUTHORIZATION_FLAGS,
PROVIDER_STORAGE_LIMIT, (which indicates allotted receiver memory);
PROVIDER_NAME, (the application provider's name)
PROVIDER_FIXED_CERT, (which is the providers public key).

This foregoing certificate information is hashed modulo 128, and the hash value is appended thereto.

The CERTIFICATE_FLAGS mentioned above include authorization flags which grant/deny the receiver processors access to privileged actions. Examples of representative privileges accessed via flags are listed immediately below and include;

1. the ability to be downloaded from a broadcast link;
2. the ability to be downloaded from a local link (e.g. connected via a local IRD port);
3. the ability to be downloaded from a local remote link (e.g. via a telephone link);
4. The ability for an application to switch tracks in the context of the same program (e.g. to switch between video tracks 1 and 2);
5. the ability to establish a broadcast connection (e.g. to change TV programs or channels);
6. the ability to establish a local connection;
7. the ability to establish a remote connection;
8. the ability to control external devices;
9. the ability to download unchecked modules;
10. the ability for an application to use cryptographic features;
11. the ability for an application to request the user for permission to access files which have user restrictions; and
12. the ability to activate a resident OCODE monitor for remote debugging.

These flags are part of both the provider certificate and the application authorization field directory. An application must have the authorization flags set at both locations before it is permitted to perform the privileged action. Respective receivers contain a BOX authorization mask, in nonvolatile storage, which is programmed to permit access to particular

which signature is made part of the respective module.

It is possible that the application provider is in fact an overseer of a subgroup of lesser application providers. In this instance, the application provider may provide a sub-certificate signed with the application providers private key and including a sub-providers public key, the sub-providers ID, an expiration date of the certificate, possibly the amount of storage allocated the sub-provider at the receiver etc. This sub-certificate will also be appended to the Directory Module along with the certificate assigned the application provider.

The preferred mode of protection is not secure in that it does not preclude prying eyes from detecting/interpreting the transmitted information. However the protection, i.e., inclusion of the certificate and hashing of the data, is advantageously simple to perform and does provide assurance that the data comes from an authorized source and that the integrity of the received data is insured, (if authenticated at the receiver).

Untrusted application providers, i.e., providers who may be careless in application generation and threaten the integrity of an AVI service, are not provided certificates to be appended their respective applications. Applications provided by untrusted providers are subjected to certification by a trusted certifying authority. The certifying authority may inspect the integrity of the application and then process the applications of the untrusted application providers for protection purposes, and ultimately pass the processed application to the system controller.

Security processing will be further described with reference to FIGURES 1 and 5. FIGURE 1 includes distinct hashing and encrypting elements 29, and 30, however it will readily be recognized, by ones skilled in the art of digital signal processing, that both functions may be performed in software executed by the μPC or digital signal processor, DSP, which may be included in the element 10. Once the application has been generated and stored {40} in the memory 11, the programmer will select/determine {41} the modules which are to be security protected. These modules will be tagged with an index (i). The Directory Module is assigned the highest index so that it will be processed last. Each module to be processed is assigned a "Change" flag which is set to a "1". The AVI system repeatedly transmits the application during an AVI program. Nominally Code and some Data Modules remain unchanged during the program, but some e.g., Data Modules may change. During the repeated transmissions of the application it is preferred not to re-process unchanged modules for security, but rather only to reprocess the modules which actually change. The "Change" flags are established to alert the security processing function which modules require reprocessing due to changes during the duration of a program. Initially the "Change" flag of each module to be security protected is set to the change mode. The element 10 also determines if a certificate from the system controller is available.

An operating index "i" is set to zero {42} and the first module, M(0), is accessed {43} from the memory 11. The "Change" flag for the module is tested {44} and reset {45}. If the module has been previously processed and the "Change" flag indicates no change, the system jumps to step {56}, increments the index "i" and accesses the next module. If the "Change" flag indicates that a change has occurred in the module, it is applied {46} to a HASH function processor 29. The particular hash function is maintained relatively simple to limit processing requirements imposed on respective receivers. The function is preferably a one-way function. The hash function should be computationally fast and extremely difficult to decipher or break. An exemplary hash function is based on a vector **W** of 256 codewords w_x each 128 bits in length. Data to be hash processed (hashed) is segmented into 256-bit exclusive blocks of data **D**, where $\mathbf{D} = d_1, d_2, d_3, d_4, d_{256}$. A basic hash function BH(**D**) is defined:

$$BH(D) = \sum_{x=1}^{256} d_x w_x \bmod 2^{128}.$$

If there are n blocks of data **D**, n values, $B_1, B_2, B_3, \ldots B_n$, each 128 bits in length, of the function BH(**D**) will be generated. To compute the hash over the whole data , the intermediate results B_i are combined as follows: Let ⟨Bi,Bj⟩represent the 256 number obtained by concatenating the 128 blocks Bi and Bj. Then define H(D), the hash over the data as;

$$H(D) = BH(\langle BH(\ldots(\langle BH(\langle BH(\langle B_1, B_2\rangle), B_3\rangle), B_4\rangle), \ldots), B_n\rangle).$$

Alternatively the function H(D) may be of the form;

$$H(D) = B_1 XOR\ B_2 XOR\ B_3\ XOR \ldots . B_n.$$

A preferred hash function for hashing modules, is the publicly known function MD5 (MD stands for Message Digest and MD5 is described by R. Rivest, THE MD5 MESSAGE DIGEST ALGORITHM, RFC 1321, April 1992). Once the module is hashed, the module is tested {47} to determine if it is expected to be changing during the program. If it is expected to change, the hash value H(D) is not placed in the directory, by rather is appended {48} to the module stored in the memory 11. (Alternatively, the hash value may be signed (encrypted) with the providers private key, and then appended to the module stored in memory 11). The index i is incremented {56} and the next module is accessed from memory. If at

controller public key has been used to generate the signature. They represent a number N from 0-15. If 0 <= N <= 14, the N'th embedded public key is to be used. If N == 15 then the public key to be used is the latest key received through an EXTERNAL trusted channel. Internally in the system the key numbers can only increase. That is, if internally the key is 5 and a certificate with key 6 occurs and is verified, then the internal key becomes 6 and certificates with keys less than 6 will not be accepted. Finally, if and when all internal keys break, the public key would be accepted only from EXTERNAL trusted channel.

The next byte is reserved for a description of the certificate. Currently it is unused. The next two bytes provide information about the Producer/Server and the Key being certified. The first byte contain flags describing the algorithm with respect to the public key; this is common to both server and producer. The next and last byte are different for Producer and Server so they are described separately.

The algorithm byte flags are:

RSA_3_WITH_MD5 (0x00008000)

RSA_WITH_MD5 (0x00004000)

The last byte for the PRODUCER certificate currently has no flags defined. The last byte for SERVER certificate currently has one flag defined to indicate whether the server is constrained. From a functional standpoint, the constrained server does not require any information about the connecting party when it first establishes a secure link. This makes link establishment much faster.

SERVER_CONSTRAINED (0x00000080)

The externally available fixed part of a Producer Signature consists of a two byte flags field which specifies the type of signature, and a 2 byte field which gives the size of the signature.

PRODUCER_SIGNATURE_FLAGS_LENGTH (2 bytes)

PRODUCER_SIGNATURE_SIZE_LENGTH (2 bytes)

Currently only one flag is meaningful, the assisted flag. If the producer's signature algorithm is RSA_3_WITH_MD5, as specified in the certificate for the producer then the producer has the option of adding additional help data after the signature for faster checking, and this is designated

PRODUCER_SIGNATURE_ASSIST (0x8000)

The Public Key Structure (for RSA) for producers, servers and OpenTV boxes will now be described. For portability, the modulus and exponent sizes MUST BE A MULTIPLE of 4 BTYES. Also OpenTv requires that if a modulus size is stated to be S bytes, then the first 32 bits of the modulus when represented in big endian format must be NON ZERO. This is not a limitation since the size can then be stated to be S-4 or less.

The public key consists of:

fixed_public._key_t,

followed by

exponent (in big-endian byte format)

followed by

modulus (in big-endian byte format)

A producer/server certificate consists of a cleartext part which contains a descriptor of the certificate by the OpenTV controller followed by the public key of the producer/server of data type described above. In addition there is an enciphered part which is the digital signature S of the OpenTV controller on data which depends on the cleartext data. A producer/server at this stage has a choice of either using only S or adding additional data (e.g., Q1 and Q2) beyond the signature to make it easier to check.

The producer/server needs to add 4 bytes of information between the Cleartext and the Signature S, and possibly some information beyond S to assist in checking. The 4 bytes of information include fields for flags and the size of the total amount of data consisting of the signature and help information.

The sizes of these two fields are:

CERTIFICATE_SIGNATURE_INFO_FLAGS_LENGTH(2 bytes)

CERTIFICATE_SIGNATURE_SIZE_LENGTH (2 bytes)

Only one flag is currently defined, the RSA_3_ASSIST flag. If set, there is help information beyond the signature S, which are the two quotients Q1, Q2 described herein above. This flag is defined;

RSA_3_ASSIST (0x8000).

The foregoing describes the state of encryption at the module level. This may be overlain with a further encryption at the transport packet level. That is, when respective modules are divided into transport packet payloads for transmission, the payloads may be encrypted independent of the authentication process.

Returning to the general description of the system, two way communications between providers and receivers, by telephone modem for example, will incorporate encrypted communications using RSA or Data Encryption Standard, DES cryptography for example. The session key will be set up using public key cryptography. An application must present a certified version of the public key of the server with which it wishes to communicate. A session key is established only if the application provider's ID matches the server ID on the certificate, and the key exchange will use the public key contained in the certificate.

However in this example, the EPROM may be reprogrammed to update system functions via an interactive transmitted program.

Assume that at manufacture, the system is programmed to look for a system maintenance SCID between 1:00 AM and 4:00 AM on mornings that the receiver is not in use, so that the system provider can update respective receivers with new system enhancements. Between 1:00 AM and 4:00 AM on mornings that the receiver is not in use, the μPC will program the SCID detector to look for packets that contain the system maintenance SCID, and prepare the memory 88 to receive program data. An example of program module detection is illustrated in FIGURE 10.

The programming for SCID detection and memory preparation is part of the start up processes {100}. Once the SCID detector is programmed the system idles {102} until a packet containing the system maintenance SCID is detected. When such packet is detected, the packet is tested {104} to determine if it contains a transmission unit or module header. If it does not, the packet is discarded and the system waits {102} for the next application packet. It is assumed that the information necessary to load any application program is self contained in the program (TU headers or Directory Module header), hence the system is constrained not to load any detected packets until a packet with the appropriate header information is available. When an appropriate packet is detected, its payload is loaded {106} in the memory 88. The system waits {108} for the next system maintenance packet, and when it is detected it is loaded {110} in the memory 88. A test {112} is made after each packet is loaded in memory to determine if a complete module has been loaded. If the module is not complete, the system jumps back to step {108} to await the next packet. If the module is complete such is noted in a listing {114}.

A further test is made {116} to determine if the completed module is a Directory Module. If it is, the system will immediately attempt to authenticate the application provider. The certificate appended to the Directory Module is decrypted {122} and its contents are checked {124}. If the contents of the certificate are not authentic, a warning display is activated to inform the user that an unauthorized provider was detected. At this point a number of alternatives are possible, including a) restarting the process at step {100}; b) shutting down the process for twenty four hours; c) discarding the Directory Module and waiting for the next Directory Module; etc. Figure 11 shows the authentication process in more detail. If at test {16} a Directory Module is detected, the certificate and encrypted hash value appended to the module are accessed {1221}. The certificate is applied to the decryptor 97 and decrypted {1222} using the AVI system controller's public key which has been previously distributed to respective receivers and stored in the receiver. The decrypted certificate is applied to the μPC {1241}. The μPC accesses corresponding items from the EPROM, and compares {1242} relevant corresponding items. For example the certificate will include an ID which is compared against a list of authorized ID's. In addition the certificate may include an expiration time and date which is compared against the current time and date etc. If the compared items check {1243} against corresponding items stored in the receiver, the application provider's public key, which was transmitted in the certificate, is applied to the decryptor and used to decrypt {1244} the encrypted hash value appended to the Directory Module, or to decrypt any other encrypted data provided by the application provider. (At this juncture, if the entire Directory Module is encrypted, it may be accessed from memory and decrypted while the application provider's public key is applied to the decryptor.) On the other hand, if the compared items prove not to be authentic or the certificate has expired, the warning is displayed {130}.

If the application provider is proven to be authentic, the Directory Module is applied to the hash function element 96 and hashed {126}, and the hash value is compared {128} in the μPC with the decrypted Directory Module hash value that was appended to the Directory Module. (In the preferred embodiment an ASCII version of some predetermined text which is associated with the system controller/provider e.g., "OpenTv(TM)", will be attached to precede respective modules before hashing so that respective hash values will equal, for example, H(OpenTv(TM)+Module). This is indicated in FIGURE 9 by the box OTV attached to the memory 88, and is meant to imply that a digital version of the text "OpenTv(TM)" , for example, may be stored in memory 88, and multiplexed with the Directory Module when it is read from the memory.) If the hash values are not identical, the Directory Module is presumed to include errors and is discarded from memory, and the fact that the module was previously loaded is erased {134} from the listing {114}, and the system returns to the step {108} to wait for the next packet.

If the hash value of the Directory Module agrees with the appended hash value, the hash values for respective program modules are retrieved {129} from the directory for use in checking the integrity of received program modules. The system jumps to step {118} which tests whether all program modules have been loaded in memory. If they have not, memory addressing for the next module is arranged {120} and the system returns {108} to await the next appropriate packet.

If the test at {118} indicates that the application program is completely stored in memory, the respective program modules are checked for transmission integrity. Respective modules are accessed {136} from memory, applied to the hash function element 96, and hashed {138}. The respective hash values are compared {140} in the μPC with the corresponding hash values transmitted in the Directory Module, or appended to the particular module under test.. If the hash values do not agree, the module is presumed to contain errors and is discarded {150, 152}, else a test is made {142} to determine if all modules have been checked. If they have all been tested, a check is made {146} to determine if a complete security checked application is resident in memory. If not, the system is returned to step {120} to start loading a new module. If the application is complete, it is executed {148}. In this example the program will instruct the μP C

9. The apparatus set forth in claim 1 further characterized by:

a source of a digital version of predetermined text;
apparatus for preceding at least one of said modules, including the Directory Module, with said digital version of predetermined text, and wherein
said hash function element is conditioned to hash said at least one module preceded with said digital version of predetermined text.

10. The apparatus set forth in claim 9 characterized in that said predetermined text is OPENTV(TM).

11. The apparatus set forth in claim 1 characterized in that said Directory Module includes a signature S calculated using a RSA algorithm with modulus N and exponent e, and includes quotients Q1, Q2 derived from the signature S by division by the modulus N, and said processor is conditioned to verify said signature S using the quotients Q1 and Q2 without performing arithmetic division.

12. A method of processing executable applications which are transmitted as modules, in multiplexed packet format, said modules including a Directory Module containing information linking respective further application modules, said Directory Module having an encrypted certificate attached which contains information about the provider of the application, and which certificate is encrypted by a system provider's private key, said method characterized by;

detecting and selecting packets including a desired application, and storing payloads of respective packets as respective modules;.
selecting a Directory Module with encrypted certificate attached;
decrypting the certificate with the system provider's public key;
Comparing information in decrypted said certificate with corresponding stored data;
hashing ones of said application modules to produce module hash values;
comparing said module hash values with corresponding module hash values transmitted in said Directory Module; and
executing an application if corresponding produced and transmitted hash values are identical and decrypted information contained in said certificate is equivalent to said corresponding stored data.

13. The method set forth in claim 12 wherein said certificate includes an application providers public key and said Directory Module has a hash value of said Directory Module attached, which hash value is encrypted with said application provider's private key, and said method further characterized by the steps of:

extracting said application provider's public key from decrypted said certificate and separating encrypted said hash value from said Directory Module;
decrypting encrypted said hash value using said application provider's public key; and
comparing decrypted said encrypted hash value with a hash value of detected said Directory Module.

14. The method set forth in claim 12 further characterized by appending a digital form of the text OpenTv(Tm) to said Directory Module prior too hashing said Directory Module, and hashing said Directory Module with said digital form of the text OpenTv(tm) appended.

15. Apparatus for transmitting executable applications characterized by;

a processor (10,11,12) for generating an executable application, and forming said application into modules containing portions of said application and a Directory Module containing information linking modules in an application;
a hash function element (29,30) for performing one-way hash functions on modules of said application to produce corresponding hash values, and cooperating with said processor for inserting said hash values in said Directory Module;
a source (10) of an application provider's public key and a certificate signed with a private key of a system controller and including, an application provider's identifier, and a time stamp related to one of the time of generation and the time of expiration of the certificate;
means (11,12) for attaching said application provider's public key and said certificate to said Directory Module; and
a transport processor (14,16) for forming a time division multiplexed signal with said modules of said application.

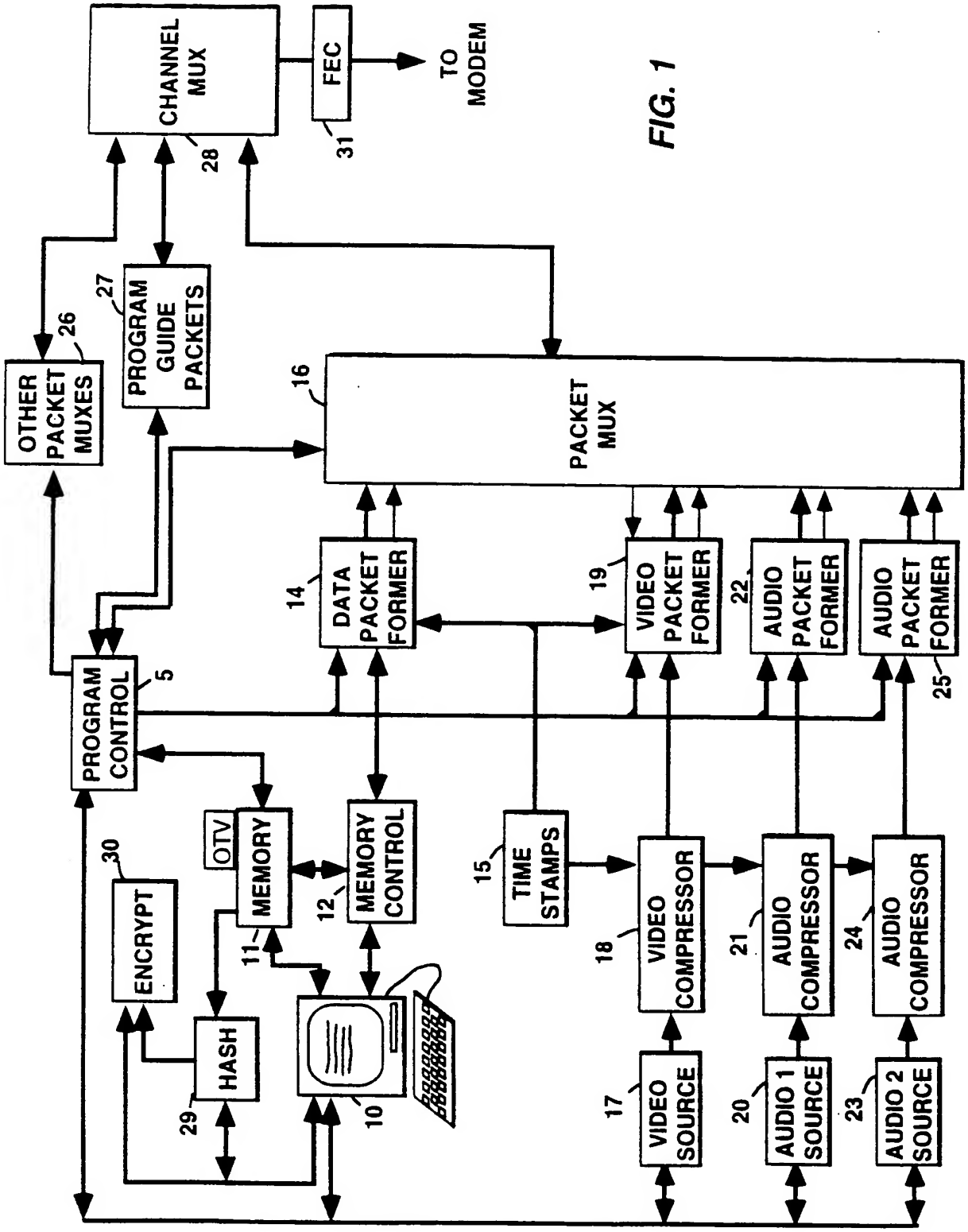

FIG. 1

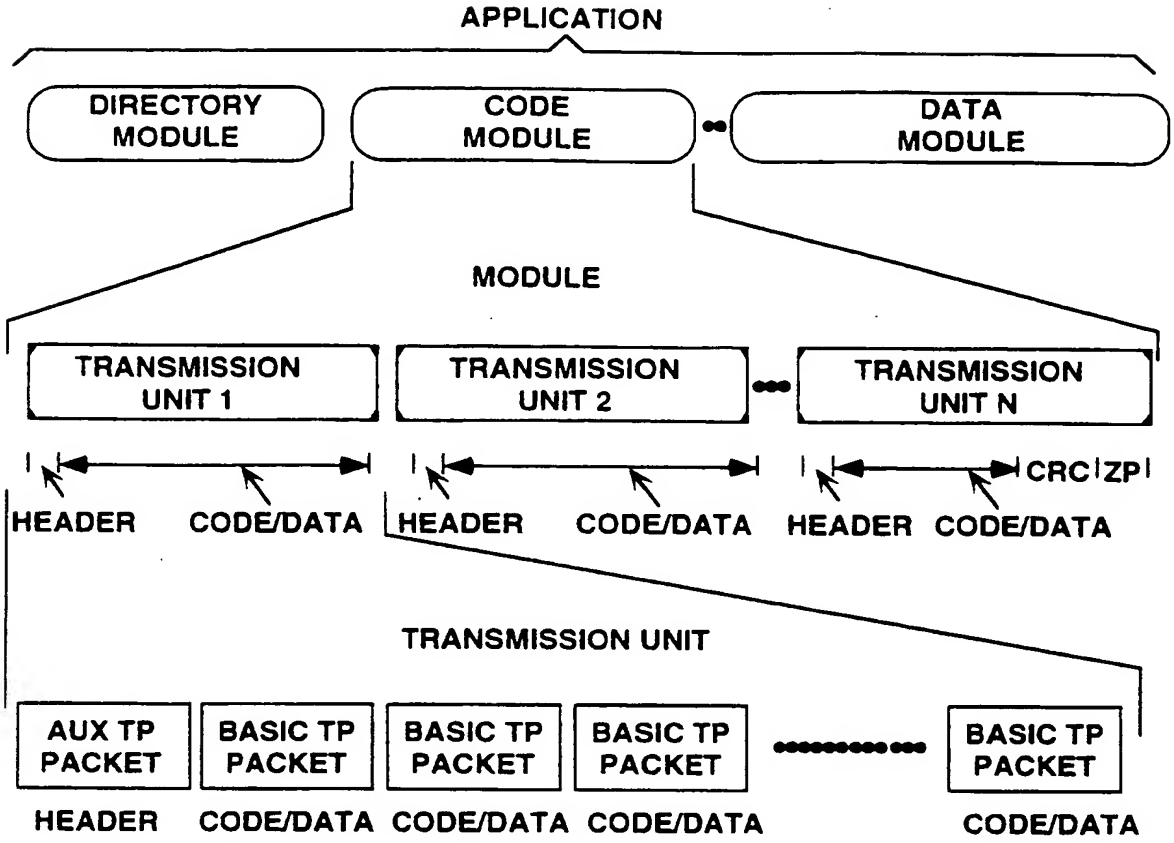

FIG. 4

TABLE I

BITS	FUNCTION
16	MODULE ID
32	TOTAL BYTES IN MODULE INCLUDING CRC
32	MODULE VERSION NUMBER
32	MODULE TRANSMISSION UNIT BYTE OFFSET
32	LENGTH (BYTES) OF TRANSMISSION UNIT
XX	RESERVED

FIG. 5

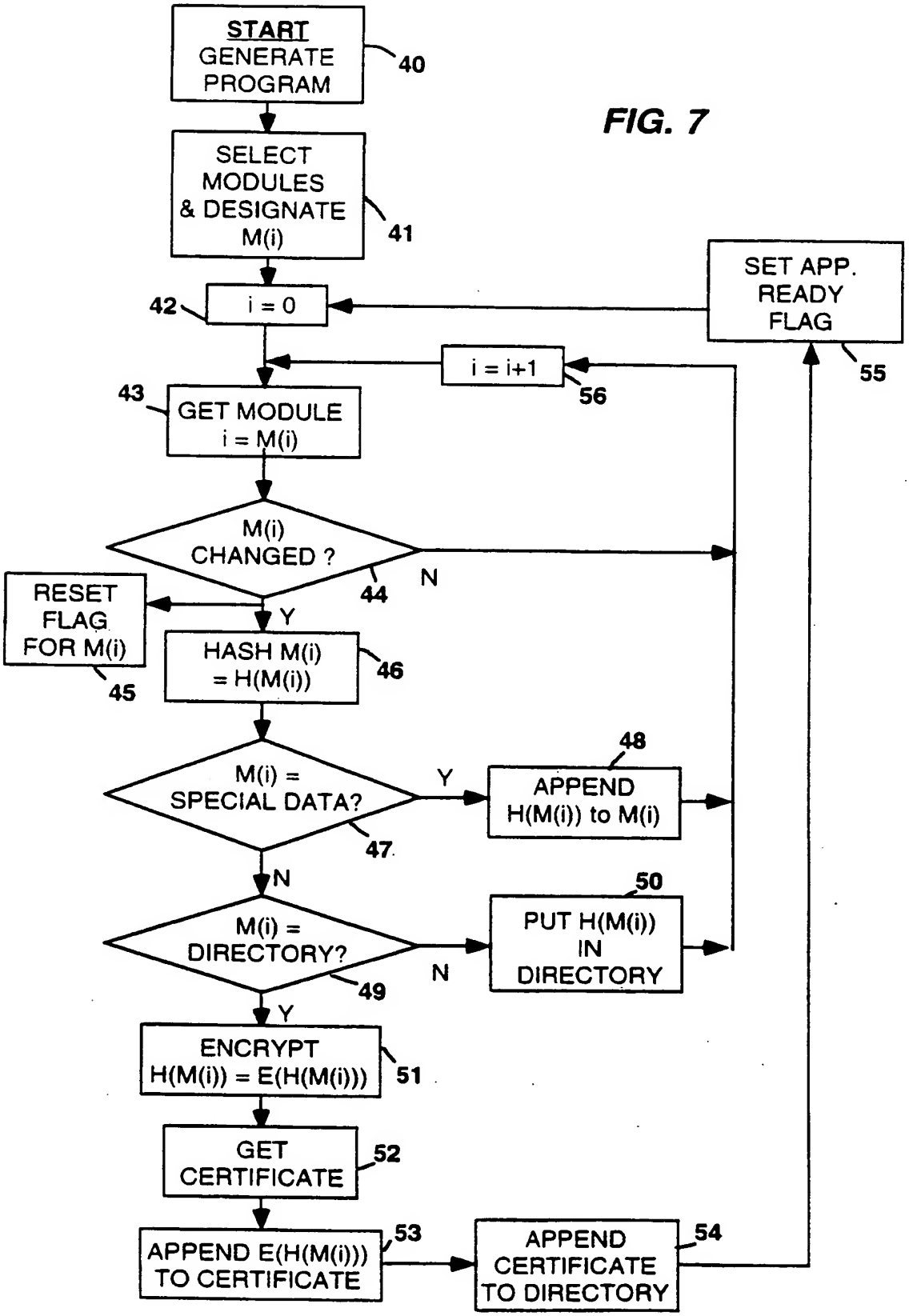

FIG. 7

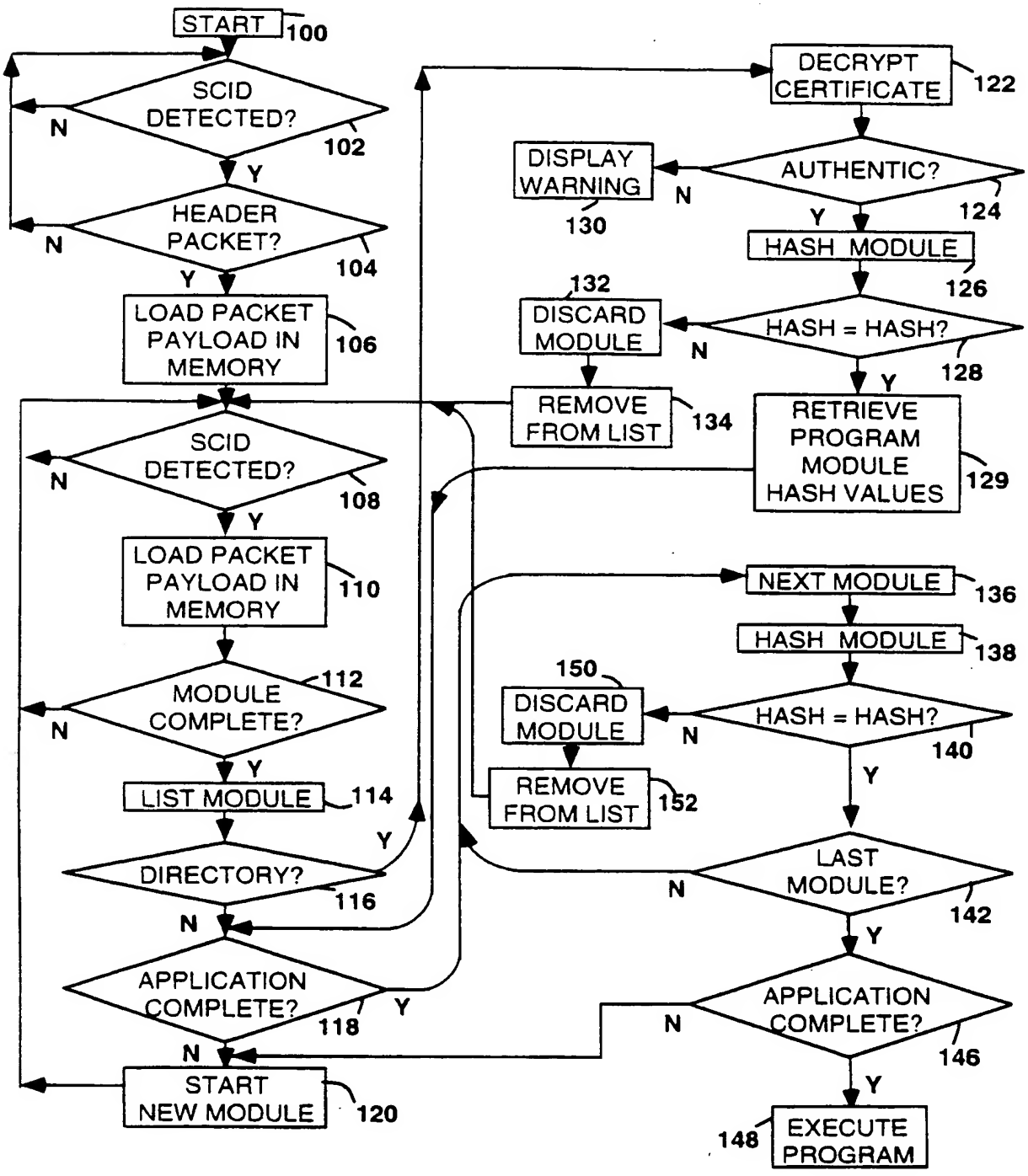

FIG. 10

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 96 11 0363

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
X	WO-A-95 15655 (SCIENTIFIC ATLANTA) 8 June 1995 * page 3, line 18 - page 4, line 26 * * page 6, line 15 - page 7, line 29 * * page 9, line 14 - page 10, line 2 * * page 14, line 19 - page 20, line 22 * * page 24, line 25 - page 25, line 30 * * page 30, line 17 - page 33, line 17 * * page 42, line 10 - line 26 * * figures 1-6,10-12 * ---	1-3,6,9, 12,13, 15-20,23	H04N7/167
X	US-A-5 420 866 (WASILEWSKI ANTHONY J) 30 May 1995 * column 1, line 39 - column 3, line 22 * * column 7, line 64 - column 18, line 5 * * figures 2-8 * -----	1-3,12, 13, 15-17, 19,20	
			TECHNICAL FIELDS SEARCHED (Int.Cl.6) H04N

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	15 October 1996	Van der Zaal, R

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)